International Journal of Recent Technology and Applied Science
p-ISSN: 2721-2017, e-ISSN: 2721-7280

Original Research Paper

# Design and Development of Effective Multi-Level Cache Memory Model

**Eze Val Hyginus Udoka[1], Eze Martin Chinweokwu[2], Edozie Enerst[3], Eze Chidinma Esther[1]**

[1] *Department of Publication and Extension, Kampala International University. Kampala, Uganda.*
[2] *Department of Electronic Engineering, University of Nigeria. Nsukka, Nigeria.*
[3] *Department of Electrical Engineering, Kampala International University, Kampala, Uganda.*

**Article history**
06.04.2023

**Revised:**
01.05.2023

**Accepted:**
09.05.2023

***Corresponding Author:**
Eze Val Hyginus Udoka
**Email:**
ezehyginusudoka@gmail.com

**This is an open access article, licensed under:** CC–BY-SA

**Abstract:** An algorithm to determine the effectiveness and efficiency of a multi-level cache was developed in this paper. The developed model was used to test the efficiency rate, the relationships and the performance output level of a computer concerning the cache properties. This research paper showed that the level of cache and access time increases with the absolute hit rate but decreases with the relative hit rate. The number of cache levels varies directly with absolute access time and inversely with relative access time. The level one cache with set associativity of one has the highest access time and as the associativity and cache levels increase the memory access time decreases. The increase in the number of set associativity leads to an increase in cache performance and as well increases the performance speed of a computer.

**Keywords:** Access Time, Associativity, Cache Memory, Hit Rate, Main Memory.

## 1. Introduction

The mismatch between the speed of microprocessor (µP) and main memory (MM) was one of the Bottlenecks that affect the performance of computers [1]. Hence, with increasing speed of the microprocessors and the decreasing speed of main memory (as the size of memory increases, the speed of the memory decreases) creates bottleneck when microprocessor accesses data from the memory [2]. Main memory is very slow because it is made from Dynamic RAM (DRAM) technology which requires intermittent refreshing. The speed differences between microprocessor and main memory became worst with the introduction of multi-core processors which led to the introduction of many memory hierarchies that takes up some percentages of the total energy consumption [1] [3] [4]. Due to the desire for high performance computer systems, cache memory is introduced between the microprocessor and the main memory to reduce the bottleneck. Cache memory is a high speed, small-sized type of volatile computer memory that provides high-speed data access to a microprocessor and stores frequently used computer programs, applications and data until a computer is restarted. Cache memory reduces the access time between the main memory and the microprocessor because it is smaller in size compared to main memory and it is made of Static RAM (SRAM) technology which requires no refreshing. The best cache configuration gives the minimum execution time and the lowest energy consumption. A quality cache configuration encapsulates total cache and block sizes, associativity, search algorithm, pre-fetch and write policies as some of the parameters that make up a good cache configuration [5] [6] [7].

A good configurable cache architecture incorporates three configurable cache parameters, configured by setting a few bits in the configuration register. Cache can be configured in software as direct mapped cache, fully associative cache and N-way set associative cache while utilizing the full capacity of cache. This type of software-based configuration is achieved by employing a technique known as way concatenation [2]. Line Concatenation technique is used to configure the cache line size. One of the most architectural configurations used to reduce the energy consumption in cache is portioning of cache into several smaller caches known as levels of caches. These levels of cache also reduce energy consumption, increases efficiency and performance by exploiting locality of reference. Most of the cache energy consumption is due to dynamic power and a small fraction is due to static power. Dynamic power consumption occurs as a result of switching activities of transistors during cache access while static comes from current leakage, even when the cache is not being accessed [8]. However, 20-50% of the energy consumption of the on-chip cache is attributed to Translation Lookaside Buffer (TLB). Translation lookaside buffer helps to translate virtual address issued by the processor to physical address present in the cache or the main memory [9] [10] [11]. Dynamic energy consumption characterized by first level (L1) caches since they are more frequently accessed than the other levels of cache, due to the parallel access of tag and data arrays. Locality of reference is a rule in computer architecture in which program tends to reuse resources that has been most frequently/recently used [12]. The amount of data transferred between the main memory and the write back operation or policy is dependent on the cache line. The larger the cache line size, the more data that is likely related are closed together and brought into the cache block at the same time. When a processor needs a word, it generates a reference address, and it will use the generated reference address to search for the world in cache block. However, if it is in the cache, it will be delivered to the processor and it is known as cache hit, but if it is not in the cache block it will be searched for in main memory and it is known as cache miss. Hit rate is the average percentage or frequent hit in cache by the processor without miss. It was observed in [12] that the memory size is directly proportional to the hit rate, the latency/access time and varies inversely to the miss rate. However, the bigger the memory size, the better the hit rate but the worst the access time. This shows the relationship between the cache levels depending on the designer's choice. The efficiency of high-performance shared memory multiprocessor depends on the design of cache coherence protocol [13] [14] [15].

The expected high performance of future computers depends on the levels of cache memory in the system to effectively curb the delay encountered my microprocessor in fetching instructions and data from memory, multiple cache memory hierarchy are employed. In current microprocessors such as Core i7, there are three levels of cache memory, thus level one (L1) cache, level two (L2) cache and level three (L3) cache. L1 cache is always highly optimized to achieve low latency on memory request hit. Every instruction fetch and every data memory reference rely on the timely response of L1 cache to keep the pipeline structure filled. L1 cache is not normally optimized for low power consumption. However, higher order cache levels (L2, L3 etc.) are optimized for moderate power savings at the cost of slightly prolonged hit latencies and marginally decreased hit rates. Saving of energy in the memory subsystem can effectively control aging effects and also extend lifetime of the cache [5]. These compromises in performance are usually not acceptable for L1 caches because of the risk of impacting overall system performance [16] [17] [18] [19].

## 2. Literature Review

This model was developed following the normal cache policy principle and memory request flow rate in a multi-level computer memory

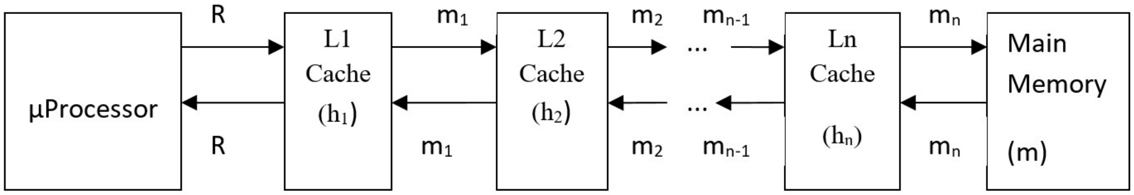

Figure 1. Memory Request flow in Multi-Level Computer Memory

From Figure 1, R is total fractional memory request. If h1 is the absolute hit rate of L1 cache, the absolute miss rate (m1) of L1 cache is given in (1).

$$m_1 = 1 - h_1 \tag{1}$$

In a multi-level cache system, the resultant hit rate ($h_{r1}$) and the resultant miss rate ($m_{r1}$) of $L_1$ are given by (2) and (3) respectively.

$$h_{r1} = h_1 \tag{2}$$

$$m_{r1} = 1 - h_1 \tag{3}$$

For the $L_2$ cache, the absolute hit rate is $h_2$ and the absolute miss rate ($m_2$) is given by (4).

$$m_2 = 1 - h_2 \tag{4}$$

In a multi-level cache system, the resultant hit rate ($h_{r2}$) of the $L_2$ cache is given by (5) while the resultant miss rate ($m_{r2}$) of the $L_2$ cache is given by (6).

$$h_{r2} = h_2(1 - h_1) \tag{5}$$

$$m_{r2} = (1 - h_1)(1 - h_2) \tag{6}$$

For the $L_3$ cache, the absolute hit rate is $h_3$ and the absolute miss rate ($m_3$) is given by (7).

$$m_3 = 1 - h_3 \tag{7}$$

In a multi-level cache system, the resultant hit rate ($h_{r3}$) of the L3 cache is given by (8) while the resultant miss rate ($m_{r3}$) of $L_3$ cache is given by (9).

$$h_{r3} = h_3(1 - h_1)(1 - h_2) \tag{8}$$

$$m_{r3} = (1 - h_1)(1 - h_2)(1 - \mathrm{h}_3) \tag{9}$$

Applying the law of mathematical induction to equations (2), (3), (5), (6), (8) and (9), equations (10) and (11) are obtained for the resultant hit rate ($h_{rn}$) and resultant miss rate ($m_{rn}$) for $L_n$ cache.

$$h_{rn} = \mathrm{h_n} \prod_{i-1}^{n} (1 - h_{i-1}) \tag{10}$$

$$m_{rn} = \prod_{i=1}^{n} (1 - h_i) \tag{11}$$

For a multi-level cache system with cache absolute access time varying from $t_1$ for $L_1$ cache to $t_n$ for $L_n$ cache and main memory absolute access time $t_m$, the total access time ($t_t$) required by the microprocessor to retrieve R request from memory and relative access time required by the processor to retrieve some request from cache memory is given by (12) and (13) respectively.

$$t_t = \sum_{i=1}^{n} h_i t_i \prod_{i=1}^{n} (1 - h_{i-1}) + \ t_m \prod_{i=1}^{n} (1 - h_i) \tag{12}$$

$$t_r = \sum_{i=1}^{n} h_i t_i \prod_{i=1}^{n} (1 - h_{i-1}) \tag{13}$$

The cache effectiveness ($E_c$) of a computer memory system with a multi-level cache system is given by (14).

$$E_c = \frac{t_m}{\sum_{i-1}^{n} h_i t_i \prod_{i-1}^{n} (1 - h_{i-1}) + \ t_m \prod_{i-1}^{n} (1 - h_i)} \tag{14}$$

The hit rate of cache memory depends on the cache capacity and the associativity of the cache. The hit rate of nth cache memory is given by (15).

$$h_n = 1 - \frac{1}{2^{\left(\frac{log_2(C_n) + log_2(A)}{10}\right)^2}} \tag{15}$$

Where A is the associativity of the cache memory and $C_n$ is the capacity of the $L_n$ cache memory in bytes. The capacity of nth cache memory is given by (16).

$$C_n = 2^{CEIL(\ log_2} \left( \sum_{i=1}^{n-1} 2^{3-i}\ C_i \right) ) \tag{16}$$

Substituting equation (16) in equation (15), equation (17) was obtained.

$$h_n = 1 - \frac{1}{2^{\left(\frac{\mathrm{CEIL}(\log_2\left(\sum_{i=1}^{n-1} 2^{3-i} C_i\right)) + log_2(A)}{10}\right)^2}} \tag{17}$$

The time $t_n$ in (14) is the absolute access time of cache memory and it is given by (18).

$$t_n = \frac{1}{7} Exp \left( \frac{CEIL(log_2(\sum_{i=1}^{n-1} 2^{3-i} C_i)) + log_2(A)}{10} \right) ns \tag{18}$$

## 3. Methodology

This model was mathematically developed following the acceptable and existing cache policy principle and memory request flow rate in a multi-level computer memory. Finally, the model was developed by applying the law of mathematical induction in equations (2), (3), (5), (6), (8) and (9) to obtain a universal resultant cache hit rate (h_rn) and resultant cache miss rate (m_rn) for Ln cache.

## 4. Finding and Discussion

This section of the paper discussed in detail the results obtained based on the level of cache capacity, associativity, cache absolute hit rate and the performance of cache levels in relation to cache capacity and associativity.

This section of the paper discussed in details the results obtained based on the level of cache capacity, associativity, cache absolute hit rate and the performance of cache levels in relation to cache capacity and associativity.

Table 1. Effects of Cache Capacity and Associativity on the Cache Absolute Hit Rate

| **No of Sets** | **Cache Capacity (kB)** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **4** | **8** | **16** | **32** | **64** | **128** | **256** | **512** | **1024** | **2048** | **4096** |
| | **Cache Absolute Hit Rate** | | | | | | | | | | | | |
| 1 | 0.50 | 0.57 | 0.63 | 0.69 | 0.74 | 0.79 | 0.83 | 0.87 | 0.89 | 0.92 | 0.94 | 0.95 | 0.97 |
| 2 | 0.57 | 0.63 | 0.69 | 0.74 | 0.79 | 0.83 | 0.87 | 0.89 | 0.92 | 0.94 | 0.95 | 0.97 | 0.97 |
| 4 | 0.63 | 0.69 | 0.74 | 0.79 | 0.83 | 0.87 | 0.89 | 0.92 | 0.94 | 0.95 | 0.97 | 0.97 | 0.98 |
| 8 | 0.69 | 0.74 | 0.79 | 0.83 | 0.87 | 0.89 | 0.92 | 0.94 | 0.95 | 0.97 | 0.97 | 0.98 | 0.99 |
| 16 | 0.74 | 0.79 | 0.83 | 0.87 | 0.89 | 0.92 | 0.94 | 0.95 | 0.97 | 0.97 | 0.98 | 0.99 | 0.99 |

From Table 1, it was observed that associativity varies directly as cache absolute Hit rate and cache capacity in kB. Cache capacity varies directly with the cache absolute Hit rate.

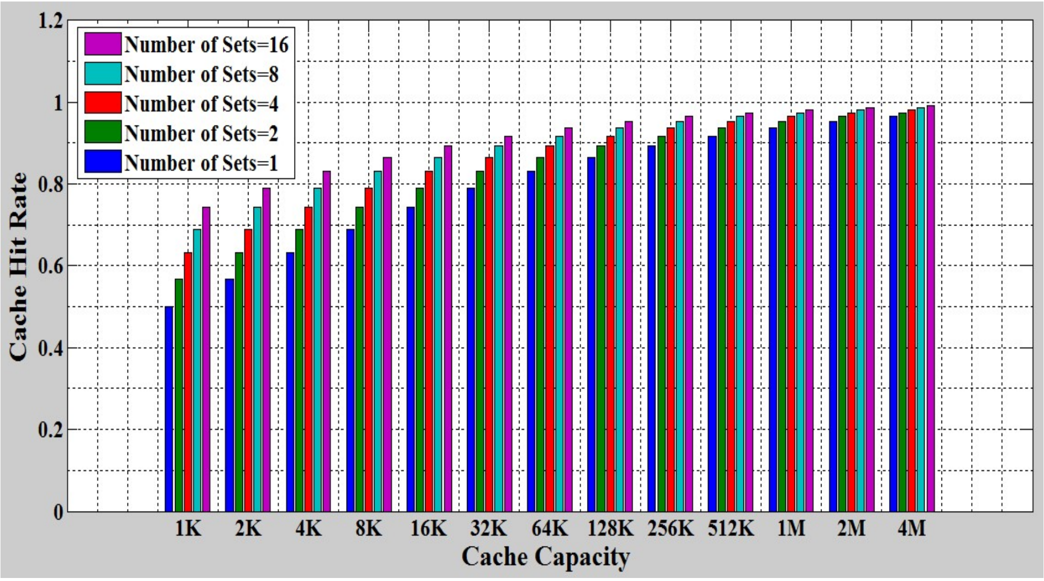

Figure 2. Effects of Cache Capacity and Cache Associativity on Cache Absolute Hit Rate

It was observed from Figure 2 that the higher the cache capacity, the higher the cache hit rate. Furthermore, an increase in associativity and cache capacity leads to an increase in hit rate though decreases speed/increases latency. It was also observed from Figure 2 that from a cache capacity of 1MB the associativity has an infinitesimal /less effect on the hit rate. This implies that the hit rate is dependent on the cache capacity but not solely on associativity.

Table 2. Effects of L1 Cache Capacity on the Capacity of L2, L3, L4, L5, L6 and L7 Caches

| **Higher Cache Level** | **L1 Cache (kB)** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **4** | **8** | **16** | **32** | **64** | **128** | **256** |
| | **Optimum Cache Capacity of Higher Order Caches(kB)** | | | | | | |
| **L2 Cache** | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |
| **L3 Cache** | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |
| **L4 Cache** | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 |
| **L5 Cache** | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 |
| **L6 Cache** | 512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768 |
| **L7 Cache** | 1024 | 2048 | 4096 | 8192 | 16384 | 32768 | 65536 |

Table 2 shows the relationship between level one cache and the other levels of cache. From equation (15), it was observed that as far as the level 1 cache is known every other level can be determined. Table 2 was generated in accordance with equation (15) and it concurs with the rule of cache placement policy.

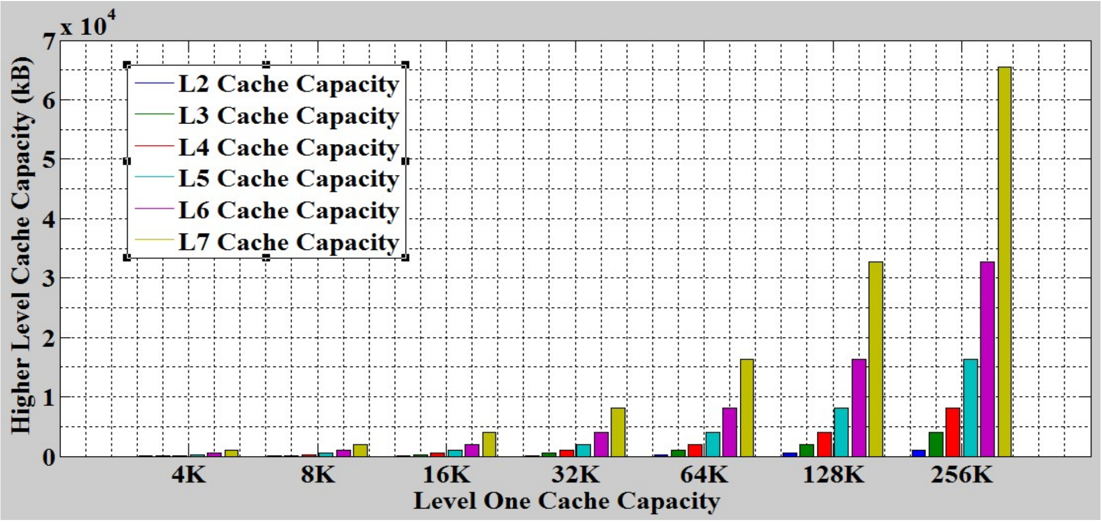

Figure 3. Plot of Effects of L1 Cache Capacity on the Capacity of L2, L3 and L4 Caches

From Figure 3, it was observed that as the cache capacity increases progressively, the cache level capacity increases. Increase in the level of cache capacity varies directly with level one cache capacity. The higher-level cache capacity increases with increase in cache capacity.

Table 3. Variation of Absolute and Relative Hit Rate with Cache Size and Cache Level Respectively

| **Cache Hit Rate** | **Cache Level** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **L1 (4KB)** | **L2 (16KB)** | **L3 (64KB)** | **L4 (128KB)** | **L5 (256KB)** | **L6 (512KB)** | **L7 (1MB)** |
| | **Number of Sets=1** | | | | | | |
| **Absolute Hit Rate** | 0.6314 | 0.743 | 0.8305 | 0.8651 | 0.8942 | 0.9181 | 0.9375 |
| **Relative Hit rate** | 0.6314 | 0.2738 | 0.0787 | 0.0139 | 0.0019 | 0.000 | 0.0000 |

Table 3 shows the relationship between absolute and relative hit rate with cache size and cache level respectively. From the table it was observed that absolute hit rate increases with increase in cache level and relative hit rate decreases with increase in level of cache.

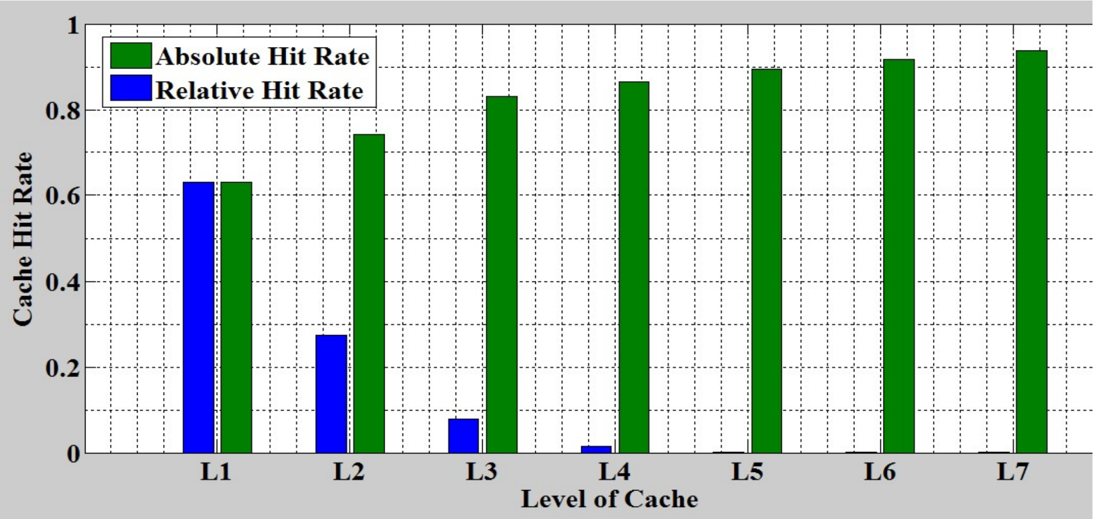

Figure 4. Variation of Absolute and Relative Hit Rate with the Cache Level

From Figure 4, Absolute hit rate varies directly to the cache levels but relative hit rate varies inversely to the cache levels. This implies that as the level of cache increase the access time increases in absolute hit rate but decreases in relative hit rate. From figure 4, it was obviously observed that relative hit rate improves the performance of a computer compare to absolute hit rate because less access time is the expectation of all designers and users of computer.

Table 4. Variation of Absolute and Relative Cache Access Time with Cache Size and Cache Level Respectively

| No of Sets | Cache Capacity (kB) | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |
| | Cache Absolute Access Time (ns) | | | | | | | | | | | | |
| **1** | 0.25 | 0.29 | 0.33 | 0.38 | 0.44 | 0.50 | 0.56 | 0.64 | 0.72 | 0.81 | 0.91 | 1.02 | 1.15 |
| **2** | 0.29 | 0.33 | 0.38 | 0.44 | 0.50 | 0.56 | 0.64 | 0.72 | 0.81 | 0.91 | 1.02 | 1.15 | 1.28 |
| **4** | 0.33 | 0.38 | 0.44 | 0.50 | 0.56 | 0.64 | 0.72 | 0.81 | 0.91 | 1.02 | 1.15 | 1.28 | 1.43 |
| **8** | 0.38 | 0.44 | 0.50 | 0.56 | 0.64 | 0.72 | 0.81 | 0.91 | 1.02 | 1.15 | 1.28 | 1.43 | 1.60 |
| **16** | 0.44 | 0.50 | 0.56 | 0.64 | 0.72 | 0.81 | 0.91 | 1.02 | 1.15 | 1.28 | 1.43 | 1.60 | 1.78 |

Table 4 shows that, as the cache capacity is increasing the cache absolute access time is increasing with respect to associativity. The access time is also increasing with increase in cache capacity. The associativity increases with increase in cache capacity in relation with cache absolute access time.

Figure 5, shows the variation of absolute and relative cache access time with cache size and cache levels. It was observed that absolute access time varies inversely as the relative access time with respect to number of cache levels. This implies that absolute access time increases with increase in number of cache level but relative access time decreases with increase in number of cache levels.

From Table 5, it was observed that as the number of set associativity increases the total memory access time decreases. Total memory access time varies inversely as the number of cache levels. This implies that a computer with higher number of set associativity and higher number of cache level gives the best performance which is the desire of every computer designer and user.

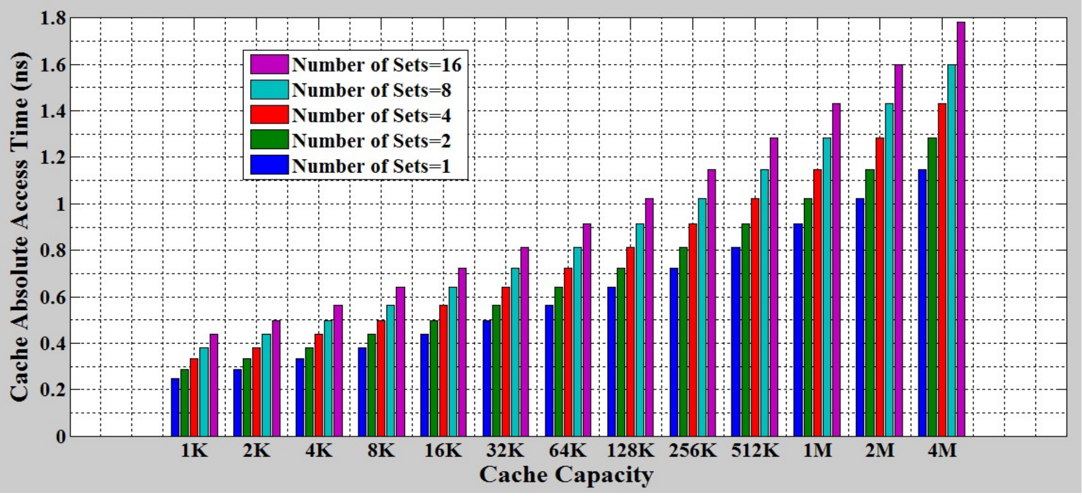

Figure 5. Variation of Absolute Cache Access Time with Cache Capacity and Cache Associativity

Table 5. Variation of total memory access time with number of cache levels for L1=4K, L2=16K, L3=64K, L4=128K, L5=256K, L6=512K and L7=1M, main memory access time =100ns

| Number of Sets | Number of Cache Levels | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | Total Memory Access Time | | | | | | |
| **1** | 37.0660 | 9.7733 | 1.9499 | 0.5699 | 0.3777 | 0.3568 | 0.3551 |
| **2** | 31.2214 | 6.8253 | 1.2201 | 0.4397 | 0.3549 | 0.3479 | 0.3474 |
| **4** | 25.9491 | 4.6756 | 0.8000 | 0.3795 | 0.3443 | 0.3421 | 0.3420 |
| **8** | 21.2842 | 3.1580 | 0.5689 | 0.3527 | 0.3389 | 0.3383 | 0.3383 |
| **16** | 17.2328 | 2.1203 | 0.4471 | 0.3410 | 0.3359 | 0.3357 | 0.3357 |

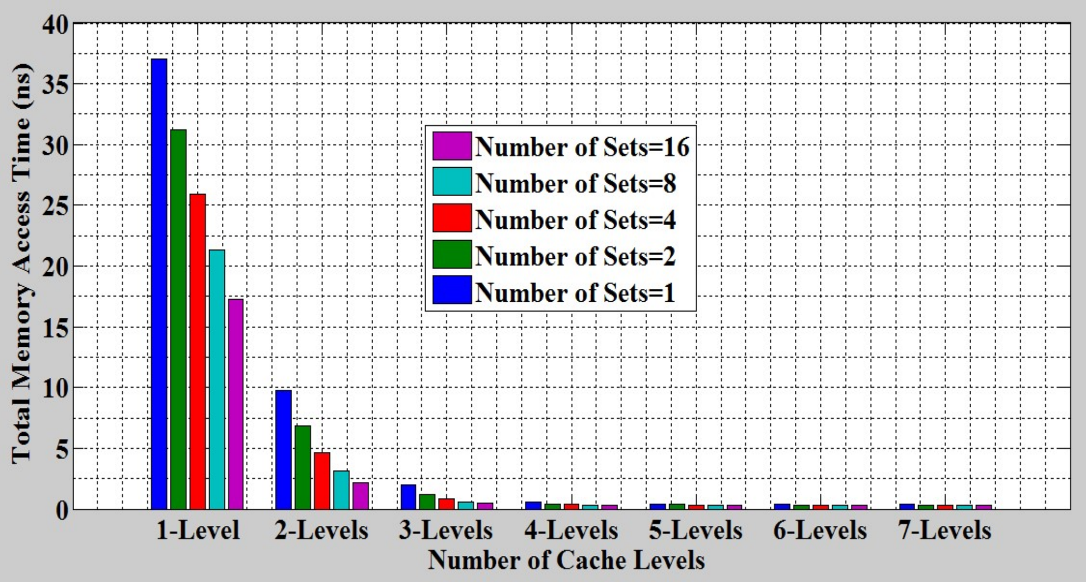

Figure 6. Variation of Relative Cache Access Time with Cache Level

From figure 6, it was observed that L1 cache to L3 cache has a very clear effect on total memory access time with respect to associativity. Furthermore, total memory access time varies inversely as the number of cache levels. From this plot, it was observed that level 1 cache and set associativity of one has the highest access time and as the associativity and cache levels increases the memory access time decreases. This plot shows that to achieve a high-performance computer, a designer should consider higher levels of cache with higher associativity.

Table 6. Variation of cache effectiveness with cache size and number of cache levels for L1=4K, L2=16K, L3=64K, L4=128K, L5=256K, L6=512K and L7=1M, main memory access time =100ns

| **Number of Sets** | **Number of Cache Levels** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| | **Total Memory Access Time** | | | | | | |
| **1** | 2.6979 | 10.2320 | 51.2842 | 175.4548 | 264.7673 | 280.2570 | 281.6336 |
| **2** | 3.2029 | 14.6515 | 81.9629 | 227.4360 | 281.7341 | 287.4684 | 287.8402 |
| **4** | 3.8537 | 21.3878 | 124.9971 | 263.5199 | 290.4129 | 292.3064 | 292.3971 |
| **8** | 4.6983 | 31.6652 | 175.7932 | 283.5656 | 295.0478 | 295.6175 | 295.6376 |
| **16** | 5.8029 | 47.1623 | 223.6673 | 293.2914 | 297.7023 | 297.8601 | 297.8641 |

Table 6 shows the variation of cache effectiveness with cache size and number of cache levels. From the table it is noticed that effectiveness of a cache depends on the set associativity and the cache level. The higher the set associativity and cache levels the more effective the cache.

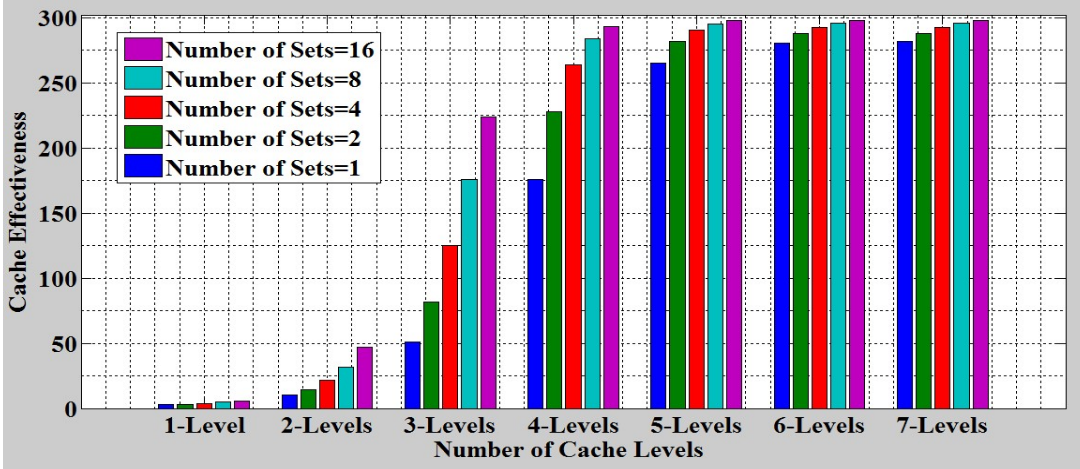

Figure 7. Variation of Cache Effectiveness with Cache Size and Number of Cache Levels

From Figure 7, it was observed that associativity and number of cache levels varies proportionally to the cache effectiveness. It shows that increase in the number of set associativity leads to increase in cache performance. It also implies that as the number of cache level increases cache effectiveness increases. In all increase in set associativity and number of cache levels increase performance of computer. The disadvantage is that it increases the complexity of the system and also cost.

## 5. Conclusion

The developed model showed the efficiency rate, relationships and the performance output level of a computer with respect to the cache properties. This research paper showed that the level of cache and access time increases with absolute hit rate but decreases with relative hit rate. The number of cache levels varies directly with absolute access time and inversely with relative access time. The level 1 cache and set associativity of 1 has the highest access time and as the associativity and cache levels increases the memory access time decreases. Finally, this model showed that, relative hit rate improves the performance of a computer, increase in the number of set associativity leads to increase in cache performance and higher levels of cache with higher associativity which also increases the performance of a computer is the earnest desire of all the computer users and designers

**Acknowledgement**
I wish to acknowledge the Department of Electronic Engineering option in software and computer Engineering, University of Nigeria Nsukka for granting us the enabling environment to carry out this research work.

## References

[1] P. Maniotis, S. Gitzenis, L. Tassiulas, and N. Pleros, “An optically-enabled chip–multiprocessor architecture using a single-level shared optical cache memory,” Optical Switching and Networking, vol. 22, pp. 54–68, 2016, doi: 10.1016/j.osn.2016.05.001.

[2] M. Jhamb, R. K. Sharma, and A. K. Gupta, “A High Level Implementation and Performance Evaluation of Level – I Asynchronous Cache on FPGA,” Journal of King Saud University - Computer and Information Sciences, 2015, doi: 10.1016/j.jksuci.2015.06.003.

[3] B. Cuesta, R. Alberto, R. Antonio, and D. Jose’, “Increasing the Effectiveness of Directory Caches by Avoiding the Tracking of Noncoherent Memory Blocks,” IEEE TRANSACTIONS ON COMPUTERS, vol. 62, no. 3, pp. 482–495, 2013.

[4] Z. Fang et al., “Reducing cache and TLB power by exploiting memory region and privilege level semantics q,” Journal of Systems Architecture, vol. 59, no. 6, pp. 279–295, 2013, doi: 10.1016/j.sysarc.2013.04.002.

[5] J. Díaz, J. L. Risco-martín, and J. M. Colmenar, “The Journal of Systems and Software Multi-objective optimization of energy consumption and execution time in a single level cache memory for embedded systems,” vol. 111, pp. 200–212, 2016, doi: 10.1016/j.jss.2015.10.012.

[6] K. Markus and W. Christian, “An overview of Cache Optimization Techniques and Cache - Aware Numerical Algorithm.”

[7] P. Maniotis, S. Gitzenis, L. Tassiulas, and N. Pleros, “An optically-enabled chip – multiprocessor architecture using a single-level shared optical cache memory,” Optical Switching and Networking, vol. 22, pp. 54–68, 2016, doi: 10.1016/j.osn.2016.05.001.

[8] J. J. Valls, A. Ros, M. E. Gómez, and J. Sahuquillo, “The Tag Filter Architecture: An energy-efficient cache and directory design,” Journal of Parallel and Distributed Computing, pp. 1–10, 2016, doi: 10.1016/j.jpdc.2016.04.016.

[9] A. Hsia, C. W. Chen, and T. J. Liu, “Energy-efficient synonym data detection and consistency for virtual cache,” Microprocessors and Microsystems, vol. 40, pp. 27–44, 2016, doi: 10.1016/j.micpro.2015.11.004.

[10] A. Hsia, C. Chen, and T. Liu, “Microprocessors and Microsystems Energy-efficient synonym data detection and consistency for virtual cache,” vol. 40, pp. 27–44, 2016, doi: 10.1016/j.micpro.2015.11.004.

[11] J. Díaz, J. L. Risco-martín, and J. M. Colmenar, “The Journal of Systems and Software Multi-objective optimization of energy consumption and execution time in a single level cache memory for embedded systems,” ELSEVIER, vol. 111, pp. 200–212, 2016, doi: 10.1016/j.jss.2015.10.012.

[12] N. Miguel and D. Cerqueira, “Cache : Why Level It,” ICCA, pp. 19–26, 2016, [Online]. Available: http://gec.di.uminho.pt/discip/minf/ac0102/0945CacheLevel.pdf

[13] B. Cuesta, A. Ros, M. E. Gmez, A. Robles, and J. Duato, “Increasing the effectiveness of directory caches by avoiding the tracking of noncoherent memory blocks,” IEEE Transactions on Computers, vol. 62, no. 3, pp. 482–495, 2013, doi: 10.1109/TC.2011.241.

[14] S. Mittal, “A survey of architectural techniques for improving cache power efficiency,” Sustainable Computing: Informatics and Systems, vol. 4, no. 1, pp. 33–43, 2014, doi: 10.1016/j.suscom.2013.11.001.

[15] H. Han, T. Alexoudi, C. Vagionas, N. Pleros, and N. Hardavellas, “Pho: A Case for Shared Optical Cache Hierarchies,” in Proceedings of the International Symposium on Low Power Electronics and Design, 2021, vol. 2021-July. doi: 10.1109/ISLPED52811.2021.9502487.

[16] Z. Fang et al., “Reducing cache and TLB power by exploiting memory region and privilege level semantics,” Journal of Systems Architecture, vol. 59, no. 6, pp. 279–295, 2013, doi: 10.1016/j.sysarc.2013.04.002.

[17] N. Laoutaris, H. Che, and I. Stavrakakis, “The LCD interconnection of LRU caches and its analysis,” Performance Evaluation, vol. 63, no. 7, pp. 609–634, 2006, doi: 10.1016/j.peva.2005.05.003.

[18] Y. Zhou, Z. Chen, and K. Li, “Second-level buffer cache management,” IEEE Transactions on Parallel and Distributed Systems, vol. 15, no. 6, pp. 505–519, 2004, doi: 10.1109/TPDS.2004.13.

[19] J. P. D. Comput, J. J. Valls, A. Ros, M. E. Gómez, and J. Sahuquillo, “The Tag Filter Architecture : An energy-efficient cache and directory design,” ELSEVIER, pp. 1–10, 2016, doi: 10.1016/j.jpdc.2016.04.016.